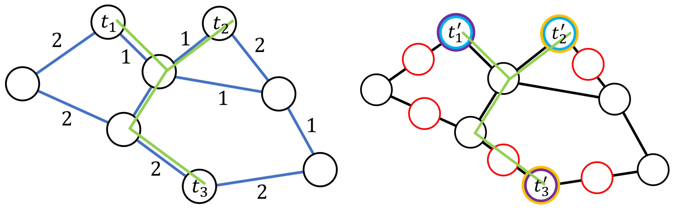

(a) RSTP instance (b) Constructed PER instance

Fig. 8. Example of NP-hard reduction.

[17] M. Bern and P. Plassmann, "The steiner problem with edge lengths 1 and 2," *Inf. Process. Lett.*, vol. 32, pp. 171–176, 1989.

Appendix A
Proof of Theorem 1

We prove the theorem by reducing the restricted case of Steiner tree problem with distances 1 and 2 (RSTP) [17] to PER. Specifically, given an undirected edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a subset of terminals $\mathcal{T} \subseteq \mathcal{V}$, where each edge $e \in \mathcal{E}$ has a cost $c_e \in \{1, 2\}$, the RSTP asks for a minimum cost subgraph where all vertices in $\mathcal{T}$ are connected and the subgraph cost is the sum of its edge costs.

We show how to construct a corresponding instance $Q = \{G = (V, E), I\}$ of the PER in polynomial time for any given instance $\mathcal{S} = \{\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{T}, c_e\}$ of the RSTP. First, we create a quantum node v and add it to V for each node in $\mathcal{V}$. Then, for each edge $e = (u, v) \in \mathcal{E}$ with $c_e = 1$, we create an edge e' to connect the corresponding quantum nodes u' and v' and add it to E. For each edge $e = (u, v) \in \mathcal{E}$ with $c_e = 2$, we create a new quantum node w and add it to V and then create two edges (u', w') and (v', w') and add them to E. Finally, for each pair (u, v) for any two terminals $u, v \in \mathcal{T}$, we create a request i with one requirement r in $R(i)$ and $D(r) = \{u', v'\}$, and add it to I; thus, there are $\binom{|\mathcal{T}|}{2}$ requests in total. Take Fig. 8 for example. For each edge $e \in \mathcal{E}$ with $c_e = 2$ in Fig. 8(a), we create a new quantum node w and its two edges, as shown in red circles in Fig. 8(b). Then, for terminal t_1, we create request i_1 with one requirement r_1 in $R(i_1)$ and $D(r_1) = \{t_1', t_2'\}$ (blue circles). Similarly, for t_2 and t_3, we create i_2 with r_2 in $R(i_2)$ and $D(r_2) = \{t_1', t_3'\}$ (purple circles) and i_3 with r_3 in $R(i_3)$ and $D(r_3) = \{t_2', t_3'\}$ (orange circles). The above reduction can be done in polynomial time.

Let $OPT_{\mathcal{S}}$ and OPT_Q be the optimal objective values of the RSTP instance $\mathcal{S}$ and the PER instance Q, respectively. We then demonstrate that $2 \cdot OPT_{\mathcal{S}} = OPT_Q$. For each edge $(u, v) \in \mathcal{E}$ with $c_e = 1$, E has a corresponding edge (u', v'). Similarly, for each edge $(u, v) \in \mathcal{E}$ with $c_e = 2$, E has two corresponding edges $(u', w'), (w', v')$ with a non-terminal vertex $w' \in V$. Thus, each edge with cost c_e in $\mathcal{E}$ has either one or two corresponding edges with cost $2c_e$ in E, indicating that each chosen edge in E takes two qubits. This construction for Q ensures that the optimal solution in G that can serve any single request in I corresponds to an optimal solution that connects all terminals in $\mathcal{T}$ for $\mathcal{S}$. Conversely, the optimal solution in $\mathcal{G}$ for $\mathcal{S}$ also corresponds to an optimal solution in G for Q. The total cost of the subgraph in E is equivalent to 2 times the cost of the subgraph in $\mathcal{E}$, ensuring that the minimum cost solution is preserved. Consequently, solving the PER leads to a solution to the RSTP, establishing the reduction.

Appendix B
Proof of Theorem 2

For ease of presentation, let x and y denote the final primal and dual solutions of ISCA-SR, respectively. Let OPT denote the optimal value of the primal and $\deg_x(S)$ represent the number of edges in solution x that exists in $\delta(S)$, i.e., $\deg_x(S) = \sum_{(u,v)\in\delta(S)} x_{uv}$, where $S \subseteq V$. To prove ISCA-SR is a 2-approximation algorithm for PER-SR, we first show that ISCA-SR gives a feasible solution for PER-SR in Lemma 1 and achieves 2-approximation in Lemma 2. Subsequently, we prove ISCA-SR can terminate in polynomial time.

Lemma 1. The solutions x and y generated by ISCA-SR are primal and dual feasible for PER-SR, respectively.

Proof. Recall that SEP iteratively adds an edge (u, v) to solution x (i.e., the initial shared state) when constraint (3b) on (u, v) becomes tight until no cut is active. At that point, no cut is unsatisfied, implying that solution x contains at least one edge in $\delta(S)$ for each cut $S \in \mathcal{C}$. Therefore, x must be a feasible solution for ILP (2a)−(2c). Then, it suffices to show that every feasible solution of ILP (2a)−(2c) must meet the requirements of PER-SR. Suppose x is a feasible solution of ILP (2a)−(2c). The solution must include the edges that connect u and v for each (u, v) with $f(u, v) = 1$ according to constraint (2b). This indicates that all terminals in each requirement in PER-SR are connected in solution x. Consequently, x is primal feasible.

When a constraint (3b) becomes tight, the corresponding edge (u, v) will be added to the solution, and each active cut S with $(u, v) \in \delta(S)$ will be inactivated. The dual variable y_S of such inactivated cuts S will not increase anymore, ensuring that the constraint (3b) in subsequent rounds would not be violated. In other words, the two nodes incident to edge (u, v) are in the same component, and there is no more active cut S such that $(u, v) \in \delta(S)$. Therefore, y is dual feasible. $\square$

Lemma 2. For the solution x generated by ISCA-SR,

$$\sum_{(u,v)\in E} 2 \cdot x_{uv} \leq 2 \cdot OPT$$

Proof. LP duality guarantees that any feasible dual solution's value in the dual objective (3a) is no greater than the optimal value of the primal, i.e., $\sum_{S\in\mathcal{C}} y_S \leq OPT$ for any feasible y. Thus, we can take any feasible $\sum_{S\in\mathcal{C}} y_S$ as a lower bound of OPT. Then, it suffices to show that the generated dual solution y ensures:

$$\sum_{(u,v)\in E} 2 \cdot x_{uv} \leq 2 \sum_{S\in\mathcal{C}} y_S$$

Since the dual constraint (3b) of each added edge is tight,

$$\sum_{(u,v)\in E} 2 \cdot x_{uv} = \sum_{(u,v)\in E} x_{uv} \left(\sum_{S\in\mathcal{C} | (u,v)\in\delta(S)} y_S \right)$$

Then, by changing the order of summation, we derive

$$\begin{aligned}
&\sum_{(u,v)\in E} x_{uv} \left(\sum_{S\in\mathcal{C} \mid (u,v)\in\delta(S)} y_S \right) \\
= &\sum_{(u,v)\in E} \left(\sum_{S\in\mathcal{C} \mid (u,v)\in\delta(S)} x_{uv}\cdot y_S \right) \\
= &\sum_{S\subseteq\mathcal{C}} \left(\sum_{(u,v)\in\delta(S)} x_{uv}\cdot y_S \right) \\
= &\sum_{S\subseteq\mathcal{C}} y_S \left(\sum_{(u,v)\in\delta(S)} x_{uv} \right) \\
= &\sum_{S\subseteq\mathcal{C}} y_S \cdot \deg_x(S)
\end{aligned}$$

Next, it suffices to show:

$$\sum_{S\subseteq C} \deg_x(S)\cdot y_S \le 2\sum_{S\in\mathcal{C}} y_S. \tag{4}$$

We will prove that in each iteration in SEP, the increase in the left-hand side of inequality (4) is bounded by that in the right-hand side. Recall that PER-SR increase the dual variable y_S for each active cut S at a *uniform* rate until some edge's constraint in (3b) becomes tight. Consider the iteration at which the t-th edge is added by PER-SR, i.e., iteration t, and let Λ_t be the increase in dual variable y_S for each active cut S during iteration t. In addition, let $\mathcal{C}_t$ denote the collection of active cuts in iteration t. Our goal is to show that:

$$\Lambda_t \times \left(\sum_{S\in\mathcal{C}_t} \deg_x(S) \right) \le 2\Lambda_t \times |\mathcal{C}_t|$$

That is, it suffices to show that:

$$\frac{\sum_{S\in\mathcal{C}_t} \deg_x(S)}{|\mathcal{C}_t|} \le 2. \tag{5}$$

In other words, we need to show that the average degree per active cut with respect to solution x is at most 2 in this iteration.

Let H be a graph consisting of the node set V and an edge set E', where E' includes edges (u,v) with $x_{uv}=1$ in x. Note that H could be a tree or a forest. At the beginning of the current iteration t, we first build an auxiliary graph H' for the following proof. The idea is to shrink each connected component in H, merged in the current solution, to a single node to form a new graph H' and then calculate the average degree per shrunk node in H'. The degree of each node in H' is equivalent to the degree of its corresponding node or component in H.

For ease of reading, we call the node representing an active component (i.e., active cut) in H as *active node*, all the other nodes in H' are called *inactive* nodes. Each active node in H' has a nonzero degree since there must be at least an incident edge in solution x to satisfy its requirement, which has at least two terminals. Next, we remove all isolated nodes from H'. The remaining graph shows a forest with an average degree of at most 2. This is because the forest with $|V'|$ nodes includes at most $|V'|-1$ edges, indicating that the average node degree of the forest is at most $\frac{2(|V'|-1)}{|V'|} \le 2$.

It suffices to prove that the average degree per active node in this forest is at most 2 in iteration t. We prove it by showing that each inactive node has a degree of at least 2.

Let $\mathbb{C}$ be an arbitrary component with $F(\mathbb{C})=0$ in H' in iteration t. We aim to prove that $\deg_x(\mathbb{C}) \ne 1$ by contradiction. Assume that $deg_x(\mathbb{C})=1$, and let e be the unique edge of x connecting $\mathbb{C}$ and $\bar{\mathbb{C}}$. Since every edge in x is non-redundant, e is non-redundant. Therefore, there is a pair of nodes, say u,v, such that $f(u,v)=1$ and e lies on the unique path between u and v in E'. Since this path crosses connecting $\mathbb{C}$ and $\bar{\mathbb{C}}$ exactly once, one of these nodes must lie in $\mathbb{C}$ and the other must be in $\bar{\mathbb{C}}$. Now, since $f(u,v)=1$, we get that $F(\mathbb{C})=1$, leading to a contradiction. Besides, $deg_x(\mathbb{C}) \ne 0$ since $\mathbb{C}$ is not isolated; otherwise, it would be removed in H'.

Therefore, Eq. (5) holds, and the lemma follows. □

Lemma 3. ISCA-SR can terminate in polynomial time.

Proof. We analyze the time complexity of the proposed algorithm ISCA-SR by examining its three phases one by one.

1) State Initialization Phase (SIP): Initializing the primal variable x_{uv} for each edge $(u,v)\in E$ and the dual variable y_S for every cut S in $\mathcal{C}$ does not take time in practice since we do not need to create those variables initially. It suffices to create the variable when its value starts to increase (i.e., nonzero).

2) State Expansion Phase (SEP): In this phase, we iteratively raise the dual variables and add edges to the solution. For each iteration, SEP simultaneously raises at most $|V|$ *active* cut $S\in\mathcal{C}$ uniformly until an edge (u,v) becomes tight. To this end, it examines each edge, finds the one that would be saturated first (i.e., its constraint becomes tight), and sets its corresponding variable x_{uv} to 1. Then, it updates the variable y_S for each active cut $S\in\mathcal{C}$ and inactivates and activates some cuts in $\mathcal{C}$. Moreover, it proceeds at most $|E|$ iterations since there are $|E|$ edges. Therefore, the time complexity is $O(|E|^2)$.

3) State Pruning Phase (SPP): This phase removes unnecessary edges from the solution. For each edge (u,v) in the solution (i.e., $x_{uv}=1$), SPP checks whether setting x_uv back to zero still maintains the feasibility of the solution. The feasibility check for removing each edge involves traversing all vertices and edges, taking $O(|V|+|E|)$ time. Since there can be at most $|E|$ edges in the solution, the time complexity for SPP is $O(|E|\cdot(|V|+|E|))$.

Combining the complexities of all three phases, the total time complexity of ISCA-SR is $O(|E|\cdot(|V|+|E|))$. □

Thus, by Lemmas 1, 2, and 3, we know that ISCA-SR is a 2-approximation algorithm for PER-SR. The theorem follows.

Appendix C
Proof of Theorem 3

To prove Theorem 3, we have to guarantee the gap between the solution of each group and the optimal solution will not exceed 2. Let OPT be the optimal solution of PER, and $OPT_{\mathcal{G}}$

be the optimal solution of any group when ISCA terminates. Since any group is the subproblem of PER, the optimal solution of PER is no less than any group's optimal solution, i.e., $OPT \geq OPT_{\mathcal{G}}$. Therefore, we know that the cost of each group is at most $2 \cdot OPT$. Since we have k groups, the total cost will be no greater than $2k \cdot OPT$.

On the other hand, we have to analyze the time complexity of ISCA. To group the requirements of each request, ISCA runs ISCA-SR to calculate the cost of assigning a requirement to a group. Since there can be at most k groups, ISCA-SR is executed $k \cdot |I| \cdot k$ times in the worst case. Thus, the overall time complexity of ISCA is $O(k^2 \cdot |I| \cdot |E| \cdot (|V| + |E|))$.

With the above ratio and time complexity analyses, we know that ISCA is a $2k$-approximation algorithm. The theorem holds.